DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 1
Primeras prácticas con MATLAB

Ejecutar los siguientes comandos, trabajando de forma interactiva con Matlab. Anota a la derecha lo que hace cada una de las instrucciones que siguen.

Práctica 1. *Tablas y gráficos.*

```
i)
t=1:10
t=t+5
2*t
1./t
t=[1,t]
t=[t,100]
s=[t;  1./t]
s=[s,[0;0]]
t.^2
s.^2
ii)
t=0:0.1:4*pi;
s=linspace(0,4*pi,1000)
x=sin(t);
y=cos(t)
z=exp(-t)
plot(t,x)
plot(t,y)
plot(t,x,'r',t,y,'g-*')
plot(x,y)
hold on
plot(9*x,4*y)
hold off
plot3(x,y,z)
u=x.*z
figure(2),plot(t,u)
iii)
close all
A=[t;x;y]
size(A)
size(A,1)
size(A,2)
plot3(A(1,:),A(2,:),A(3,:))
j=size(A,2)-8:size(A,2)
C=A(:,j)
```

Práctica 2. *Matrices*

```
i)
clear A
A=[0,-1,4;4,2,1/3;-1,1/2,-2]
det(A)
eig(A)
[V,D]=eig(A)
B=V'
C=V.'
D=B-C
p=poly(A)
roots(p)
ii)
B=[A(3,:);A(1,:)+A(2,:)]
v=[1,-1,0]
y=B*v
v=v'
y=B*v
z=A\v
A*z-v
```

Práctica 3. *Manipulación de funciones y representaciones graficas 2D*

```
clear all
f=@(t) sin(t)./t
ezplot(f)
fplot(f, [-20,20])
t1=pi;
y1=feval(f,t1)
t=-10:0.1:10;
y=feval(f,t);
plot(t,y)
hold on
plot(t1,y1,'dr','LineWidth',4)
title(['La funci\'on f=1-x^2 con t_1=',num2str(t1)])
xlabel('t')
ylabel('f')
hold off
```

Práctica 4. *Representaciones graficas 3D*

```
x = -7.5: .5:7.5;
y = x
[X, Y] = meshgrid (x , y);
R = sqrt (X.^2 + Y.^2) + eps;
Z = sin (R)./ R ;
mesh(x, y, Z)
surf(x, y, Z)
```

Práctica 5. *Escribir en un fichero instrucciones de MATLAB que, utilizando el Algoritmo de Newton, calculen aproximaciones a un cero de la función* $f(x) = x - x^3$.

```
function x=minewton(f,df,x0,TOL)
% metodo de Newton para ecuaciones, dadas f  y su derivada df

% variables de entrada: f,df ficheros .m
% x0 dato inicial vector columna
% TOL tolerancia error

% variables de salida:

x aproximacion al punto fijo en 'i=contador' iteraciones
ERR=TOL+1;
x=x0;
contador=0;
while ERR>TOL
    contador=contador+1;
    DF=feval(df,x);
    F=feval(f,x);
    w= DF\F;
    ERR=norm(w,inf);
    x=x-w;
    if contador>1000
        break
        disp('excede numero maximo de iteraciones')
    end
end
```

Llamar a este fichero **minewton.m**. *Ahora podemos ejecutarlo. Para ello guardamos las variables de entrada, la llamada al programa* **minewton.m**, *la evaluación de la función en el cero obtenido y su representación gráfica en un fichero* **script** *de nombre* **testminewton.m**.

```
clear all
close all
f=@(x) x-x.^3;   df=@(x) 1-3*x.^2; x0=-2;TOL=1e-10;

x=minewton(f,df,x0,TOL);

xx=x-1:.01:x+1;
ff=feval(f,xx);
plot(xx,ff,'b')
hold on
x1=x;f1=feval(f,x1);
plot(x1,f1,'dr','LineWidth',4)
title('El cero de f')
axis([x-1 x+1 min(ff) max(ff)])
xlabel('x')
ylabel('f')
```

```
grid on
hold off
```

Alternativamente podemos definir un fichero .m con la correspondiente función f y otro con su derivada, df, como una matriz si f fuera de varias variables. Para ello escribimos en dos ficheros:

```
function F=fun1(x)
%
F=x-x.^3;
```

```
function DF=dfun1(x)
%
DF=1-3*x.^2;
```

y los guardamos con el nombre `fun1.m` *y* `dfun1.m` *respectivamente. Esto nos permitirá evaluar la función y su derivada para cada x. A continuación escribimos las variables de entrada y llamamos al programa* **minewton.m** *en el fichero* **testminewton.m**

```
f=@fun1;  df=@dfun1;x0=-2;TOL=1e-10;
x=minewton(f,df,x0,TOL);
```

Una tercera posibilidad, es escribir los datos, la llamada al programa, las salidas gráficas y las funciones todo junto en un único fichero **function** *de nombre* **testminewton2.m** *del siguiente modo:*

```
function x=testminewton2
clear all
close all

f=@fun1;  df=@dfun1;  x0=-2;TOL=1e-10;

x=minewton(f,df,x0,TOL);

xx=x-1:.01:x+1;
ff=feval(f,xx);
plot(xx,ff,'b','LineWidth',2)
hold on
x1=x;f1=feval(f,x1);
plot(x1,f1,'dr','LineWidth',5)
% 'MarkerEdgeColor','k',...
%                   'MarkerFaceColor','g',...
%                   'MarkerSize',10)
title('El cero de f')
axis([x-1 x+1 min(ff) max(ff)])
xlabel('x')
ylabel('u')
grid on
hold off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function f=fun1(x)
f=x-x.^3;

function df=dfun1(x)
df=  1-3*x.^2;
```

Práctica 6. *Calcule los ceros de las siguientes funciones utilizando el método de Newton*
i) $f(x) = x - x^2$,
ii) $f(x) = xe^{-x} + 1$,
iii) $f(x_1, x_2) = (x_1 - x_2^2, \sin(x_1) - 0{,}7x_2)$.
iv) $f(x_1, x_2, x_3) = (3x_1 - cos(x_2x_3) - 1/2, 4x_1^2 - 625x_2^2 + 2x_2 - 1, e^{-x_1x_2} + 20x_3 + (10\pi - 3)/3)$.

Práctica 7. *Escribir el fichero función* **micuadgauss.m** *que dado el número de puntos de colocación (*$N = 4, 5, 6, \cdots$*) calcule* r*, los puntos de colocación de Gauss y los pesos* w *del método de cuadratura de Gauss en el intervalo* $[0, 1]$.

```
function [r,w]=micuadgauss(N)

%w pesos
%r puntos de colocacion de una formula de cuadratura de Gauss en (0,1)
%a partir de cuadraturas en (-1,1)

beta = .5./sqrt(1-(2*(1:N-1)).^(-2));
T = diag(beta,1) + diag(beta,-1);
p=poly(T);
r=roots(p);        %N ceros en (-1,1)
r=sort(r);

%pesos en (-1,1)

for k=1:N
    A(k,:)=r.^(k-1);
end
B=(1-(-1).^(1:N))./(1:N);
B=B(:);
w=A\B;

r=(1+r)/2;              %N ceros en (0,1)
w=w/2;                  %pesos en (0,1)
```

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 2
Métodos de Colocación

Práctica 8. *Escribir el fichero función* **legendregaussradau.m** *que dado el número de puntos de colocación ($N = 4, 5, 6, \cdots$) y la geometría de la ecuación (plana, cilíndrica o esférica) calcule los puntos de colocación de Gauss-Radau del método de colocación ortogonal.* **Indicación** *utiliza los comandos* `poly, polyval, roots, sort`*, sabiendo que*
`poly (A)`*=det*$(xI - A) = [c_1, \cdots c_{N+1}]$ *donde* $p(x) = c_1 x^N + \cdots c_N x + c_{N+1}$,
que `polyval(p,1)` *evalúa el polinomio* p *en el punto* 1,
y que `roots(p)` *calcula los ceros de* p

Práctica 9. *Escribir el fichero función* **legendregausslobato.m** *que dado el número de puntos de colocación ($N = 4, 5, 6, \cdots$), calcule los puntos de colocación de Gauss-Lobato del método de colocación ortogonal.* **Indicación** *utiliza el comando* `polyder`*. Utiliza el Help de MATLAB*

Práctica 10. *Escribir el fichero función* **mismatricesradau.m** *que dado el número de puntos de colocación de Gauss-Radau ($N = 4, 5, 6, \cdots$) y los puntos de colocación r calcule las matrices Q, $P = Q^{-1}$, A y B del método de colocación ortogonal.* **Indicación** *efectúa operaciones por columnas.*

Práctica 11. *Escribir el fichero función* **mismatriceslobato.m** *que dado el número de puntos de colocación de Gauss-Lobato ($N = 4, 5, 6, \cdots$) y los puntos de colocación r calcule las matrices Q, $P = Q^{-1}$, A y B del método de colocación ortogonal.*

Práctica 12. *Escribir el fichero función* **micolradau.m** *que implemente el método de colocación ortogonal para los problemas*

$$\begin{cases} -\frac{1}{r^d}(r^d u_r)_r = f(r), \quad r \in (0,1), \\ u_r(0) = 0 \\ u'(1) + \beta u(1) = b, \ (CC = 1) \qquad \textit{o bien } u(1) = b, \ (CC = 2). \end{cases}$$

con $d = 0, 1, 2$.

Los datos de entrada del fichero función son: d (la dimensión del problema), N (el número de puntos del método de colocación), la función f, los datos de contorno b, β, CC. Las variables de salida deben incluir los puntos de colocación r, la matriz P y los valores u.

Práctica 13. *Escribir el fichero script* **testmicolradau.m** *que ejecute el algoritmo* **micolradau.m** *para los ejemplos de la* **Practica 14** *y pinte en una gráfica la solución obtenida.* **Indicación** *utiliza el comando* `flipud(c)` *que cambia el orden de las componentes de un vector columna, de* $[c_1, \cdots, c_{N+1}]$ *a* $[c_{N+1}, \cdots, c_1]$, *el comando* `c=[zeros(1,N+1);c']` *y el comando* `c=c(:)` *'que cambia una matriz en un vector.*

Práctica 14. **(i) La ecuación*

$$\begin{cases} -\dfrac{d^2u}{dx^2} = \dfrac{\pi^2}{16}\cos\left(\dfrac{\pi x}{4}\right) \\ u'(0) = u'(1) + \frac{\pi}{4}u(1) = 0 \end{cases}$$

tiene por solución la función

$$u^{ex}(x) = \cos\left(\frac{\pi x}{4}\right).$$

Dibuja la diferencia, en valor absoluto, entre la solución exacta y la solución aproximada.

(iii) La ecuación

$$\begin{cases} -\dfrac{d^2u}{dx^2} = \dfrac{\pi^2}{4}\cos\left(\dfrac{\pi x}{2}\right) \\ u'(0) = u(1) = 0 \end{cases}$$

tiene por solución la función

$$u^{ex}(x) = \cos\left(\frac{\pi x}{2}\right).$$

Repite el mismo trabajo.

(ii) La ecuación

$$\begin{cases} -u'' - \dfrac{d}{x}u' = \left[\left(\dfrac{\pi}{4}\right)^2 x^2 - 2(d+1)\right]\cos\left(\dfrac{\pi x}{4}\right) + (d+4)\dfrac{\pi}{4}x\sin\left(\dfrac{\pi x}{4}\right) \\ u'(0) = 0, \qquad u'(1) + \frac{\pi}{4}u(1) = \sqrt{2} \end{cases}$$

tiene por solución la función

$$u^{ex}(x) = x^2\cos\left(\frac{\pi x}{4}\right).$$

Repite el mismo trabajo para distintos valores de d.

Práctica 15. *Crear el fichero* **testerrorcolradau.m** *de forma que utilizando el programa* **micolradau.m** *y para valores de N desde 4 hasta 8, calcule la solución de la* **Practica 14** *por el método de colocación con ese valor de N, que llamaremos $u_N^{ap}(x)$. Calcula los errores cometidos por el método de colocación ortogonal, calculando para esto los valores*

$$\varepsilon_N = \text{máx}\{|u^{ex}(x_i) - u_N^{ap}(x_i)|,\ i = 0, 1, \ldots N\}.$$

Dibujar la función $\frac{1}{N^2} \to \varepsilon_N$. ¿Qué conclusión puedes sacar?.

Práctica 16. *Escribir el fichero función* **micollobato.m** *que implemente el método de colocación ortogonal para el problema*

$$\begin{cases} -u_{xx} + p(x)u_x + q(x)u = f(x), \quad x \in (-1, 1), \\ u(-1) = a \\ u'(1) + \beta u(1) = b,\ (CC = 1) \qquad \textit{o bien } u(1) = b,\ (CC = 2). \end{cases}$$

Los datos de entrada del fichero función son: las funciones p, q, f, N (el número de puntos del método de colocación), los datos de contorno a, b, β, CC.

Práctica 17. *Considerar la siguiente ecuación*

$$\begin{cases} -\dfrac{d^2u}{dx^2} = \pi^2\,\text{sen}(\pi x) \\ u(-1) = u(1) = 0 \end{cases} \tag{1}$$

Se pide lo siguiente:

1. *Escribir el fichero script de nombre* **testmicollobato.m** *que calcule la solución aproximada de* (1) *para $N = 10$, con el algoritmo* **micollobato.m**,
2. *dibuje la solución aproximada de* (1)*, y*

3. *dibuje en otra ventana la diferencia entre la solución aproximada y la solución exacta* $u^{ex} = u^{ex}(x)$*, siendo*

$$u^{ex}(x) = \mathrm{sen}(\pi x)$$

Práctica 18. *Escribir el fichero función* **micolradaunol.m** *que implemente el método de colocación ortogonal para los problemas*

$$\begin{cases} -\frac{1}{r^d}(r^d u_r)_r = f(u), \quad r \in (0,1), \\ u_r(0) = 0 \\ u_r(1) = g(u(1)). \end{cases}$$

con $d = 0, 1, 2$.

Los datos de entrada del fichero función son: d *(la dimensión del problema),* N *(el número de puntos del método de colocación), las funciones* f*,* g *y sus derivadas (que debe estar en un fichero función), la tolerancia del método de Newton TOL, la condición inicial del método de Newton* v *y el dato de contorno* b*. Los datos de salida deben ser los puntos de colocación* x *y los valores de la solución en los puntos de colocación* u*.*

Práctica 19. *Escribir el fichero script* **testmicolradaunol.m** *que ejecute* **micolradaunl.m** *y pinte en una gráfica la solución (o soluciones) del problema de contorno para las ecuaciones de las* **Prácticas 20, 21, 22**.

Práctica 20. *Resolver numéricamente el problema de contorno*

$$\frac{1}{r^d}(r^d u')' = \tau^2 \frac{u^m}{1+ku^n}, \quad r \in (0,1); \qquad u'(0) = 0,\ u(1) = 1. \tag{2}$$

para $d = 0, 1, 2$*. Otras condiciones de contorno habituales son*

$$u'(1) + Bi_m u(1) = Bi_m, \tag{3}$$

donde Bi_m *es el número de Biot de transferencia de masa. Este problema corresponde a la reacción-difusión dentro de la partícula del catalizador poroso, en una ecuación adimensionalizada;* $u = u(r)$ *representa la concentración de un reactante que desaparece a una velocidad* $-f(u) = \tau^2 \frac{u^m}{1+ku^n}$ *por unidad de volumen por unidad de tiempo de la partícula catalítica. El parámetro* τ *es el módulo de Thiele.*

(i) Escribir los ficheros tipo función **funreac1.m**, **funCCdirichlet.m**, **dfunreac1.m**, **dfunCCdirichlet.m** *con el segundo miembro de la ecuación, las condiciones de contorno de (2) y sus derivadas respectivamente. Define* $\tau,\ k,\ m, n$ *como variables globales.*

(ii) Toma $\tau = 0{,}1;\ k = 0,\ m = 1, n = 0$ *y como aproximación inicial* $u_0(r) = 1$*. Toma* $\tau = 1; 10$*, y como aproximación inicial* $u_0(r) = \dfrac{cosh(\tau r)}{cosh(\tau)}$*, y compara con las graficas en Finlayson, p. 85.*

(iii) Toma ahora $\tau = 10,\ k = 0,\ m = 2, n = 0$*, compara con las graficas en Davis, p. 121.*

(iv) Selecciona distintos valores de $\tau = 7, 10, 12$*, (cf. Davis, p. 121 y ss. para* $\tau^2 = 50 \approx 49, \tau^2 = 100, \tau^2 = 150 \approx 144$ *).*

(v) Escribir los ficheros tipo función **funCCrobin.m**, , **dfunCCrobin.m** *con las condiciones de contorno (3) y su derivada espectivamente. Define* Bi_m *como variable global.*

(vi) Toma $\tau = 5$ y selecciona distintos valores de $k \in (1, 10)$, $m = 1, 2, n = 1, 2$. Toma tambien valores de $\tau < 1$ y $\tau > 2$. Toma $Bi_m = 10$, (cf. Froment).

Práctica 21. *Resolver*

$$\begin{cases} \frac{1}{r^d}(r^d c_r)_r = \tau^2 c^n e^{\gamma-\gamma/\Theta}, \quad r \in (0,1) \\ \\ c_r(0) = 0 \quad c_r(1) + Bi_m\, c(1) = Bi_m \end{cases}$$

donde la temperatura $\Theta(r) = 1 + \alpha[c(1) - c(r)] + \alpha\delta[1 - c(1)]$. Cf. B. Finlayson "Nonlinear Analysis in Chemical Engineering" pp 82 y ss.

(i) Escribir el fichero tipo función **funreacarrhenius.m**. *Define $\tau, \gamma, n, \alpha, \delta$ y Bi_m como variables globales.*

(ii) Utilizar los valores de $d = 0$, $\tau = 1$ $\gamma = 30$, $n = 1, 2$, $\alpha = 0{,}02$, $Bi_m \to \infty$ es decir $c(1) = 1$, $\Theta(r) = 1 + \alpha[1 - c(r)]$ y como aproximación inicial $c_0(r) = \dfrac{cosh(\tau r)}{cosh(\tau)}$.

Práctica 22. *Resolver el sistema*

$$\begin{cases} \frac{1}{r^d}(r^d c_r)_r = \tau^2 R(c, \Theta), \\ \\ \frac{1}{r^d}(r^d \Theta_r)_r = -\alpha\tau^2 R(c, \Theta), \quad r \in (0,1) \\ \\ c_r(0) = \Theta_r(0) = 0 \\ c_r(1) + Bi_m\, c(1) = Bi_m, \quad \Theta_r(1) + Bi\, \Theta(1) = Bi, \end{cases}$$

donde $R(c, \Theta) = c^n e^{\gamma-\gamma/\Theta}$. Cf. B. Finlayson "Nonlinear Analysis in Chemical Engineering" pp 80 y ss.

(i) Escribir los ficheros tipo función **funreacarrhenius2.m**, **funCCdirichlet2.m** *con la ecuación y las condiciones de contorno.*

(ii) Utilizar los valores de $d = 2, \tau = 1$ $\gamma = 30$, $n = 1$, $\alpha = 0{,}02$, $Bi, Bi_m \to \infty$ es decir $c(1) = \Theta(1) = 1$, y como aproximación inicial $c_0(r) = 1$ y $\Theta_0(r) = 1$.

(iii) Utilizar ahora $\alpha = 0{,}4$. y como aproximación inicial $c_0(r) = 0$, $\Theta_0(r) = 1 + \alpha$.

(iv) Utilizar ahora $\tau = 0{,}5$. Observa que hay múltiples estados estacionarios seleccionando distintas aproximaciones iniciales $c_0(r) = 1$, $c_0(r) = 0$ y $\Theta_0(r) = 1 + \alpha[1 - c_0(r)]$. Compara con los resultados de la p. 102.

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 3
Elementos Finitos

Práctica 23. *Escribir el fichero función* **mielfin1.m** *que implemente el método de elementos finitos para los problemas*

$$\begin{cases} -(p(x)u_x)_x + q(x)u(x) = f(x), \quad x \in (0,1), \\ u(0) = a \\ u(1) = b. \end{cases} \tag{4}$$

utilizando como aproximaciones de E_{T_k} *y* b_{T_k} *lo siguiente*

$$E_{T_k} = \frac{1}{h_k} p(x_k) S_1 + h_k q(x_k) S_2,$$

$$b_{T_k} = h_k f(x_k) S_3, \qquad k = 0, \cdots N,$$

donde S_1, S_2 *y* S_3 *son las matrices básicas.*

Los datos de entrada del fichero función son: las funciones p, q, f*,* N *(el número de puntos del mallado) y los datos de contorno* a, b*. Las variables de salida deben incluir los puntos del mallado* x*, y los valores* u*.*

Práctica 24. *Escribir el fichero función* **mielfin.m** *que implemente el método de elementos finitos para los problemas*

$$\begin{cases} -(p(x)u_x)_x + q(x)u(x) = f(x), \quad x \in (0,1), \\ u(0) = a \\ u(1) = b. \end{cases} \tag{5}$$

Los datos de entrada del fichero función son: las funciones p, q, f*,* N *(el número de puntos del mallado) y los datos de contorno* a, b*. Las variables de salida deben incluir los puntos del mallado* x*, y los valores* u*.*

Práctica 25. *Modifica el fichero* **mielfin.m** *que implementa el método de elementos finitos para resolver otras condiciones de contorno*

$$\begin{cases} CC = 1: & u'(0) = 0, & u'(L) = 0, \\ CC = 2: & u'(0) = 0, & u(L) = b, \\ CC = 3: & u(0) = a, & u(L) = b, \\ CC = 4: & u(0) = a, & u'(L) = 0. \end{cases} \tag{6}$$

Incorporar a los datos de entrada la variable CC para distinguir los tipos de datos de contorno.

Práctica 26. *Considerar la siguiente ecuación*

$$\begin{cases} -\dfrac{d^2u}{dx^2} + u = \left(1 + \dfrac{1}{\pi^2}\right) \operatorname{sen}(\pi x) \\ u(0) = u(1) = 0 \end{cases} \tag{7}$$

Se pide lo siguiente:

1. *Escribir el fichero script de nombre* **testmielfin.m** *que calcule la solución aproximada de* (7) *para* $N = 10$*, con el algoritmo* **mielfin.m***,*

2. *dibuje la solución aproximada de* (7)*, y*

3. *dibuje en otra ventana la diferencia entre la solución aproximada y la solución exacta* $u^{ex} = u^{ex}(x)$*, siendo*

$$u^{ex}(x) = \frac{1}{\pi^2}\ \mathrm{sen}\,(\pi x)$$

Práctica 27. ** La ecuación*

$$\begin{cases} -u_{xx} = \mathrm{sen}(x), & x \in (0,1) \\ u'(0) = u(1) = 0 \end{cases}$$

tiene por solución la función $u^{ex}(x) = \mathrm{sen}(x) - x - \mathrm{sen}(1) + 1$.

Crear el fichero **testerrorelfin.m** *de forma que utilizando el programa* **mielfin.m** *y para valores de* N *desde 100 hasta 1000 saltando de 50 en 50 calcule la solución del problema anterior por el método de diferencias finitas con ese valor de* N*, que llamaremos* $u_N^{ap}(x)$*. Calcula los errores cometidos por el método de diferencias finitas, calculando para esto los valores*

$$e_N = \mathrm{máx}\{|u^{ex}(x_i) - u_N^{ap}(x_i)|,\ i = 0, 1, \ldots N\}.$$

Teniendo en cuenta que el paso de la discretización es $h = 1/N$*, dibujar las funciones* $h \to e_N$ *y* $log(h) \to \log(e_N)$*. ¿Qué conclusión se puede alcanzar?.*

Práctica 28. *Escribir el fichero función* **mielfinC1.m** *que implemente el método de elementos finitos de clase* C^1 *para los problemas* (4) *Los datos de entrada del fichero función son: las funciones* p, q, f*,* N *(el número de puntos del mallado) y los datos de contorno* a, b*. Las variables de salida deben incluir los puntos del mallado* x*, y los valores* u*.*

Práctica 29. *Modificar el fichero* **mielfinC1.m** *que implemente el método de elementos finitos de clase* C^1 *para resolver otras condiciones de contorno dadas por* (6)

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 4
Métodos de Colocación para problemas de evolución no lineales

Práctica 30. *Crear el fichero tipo función* **micolevolradaunol.m** *que tomando como datos la función f, N (número de puntos de colocación interiores), T, N_t (tamaño de la discretización temporal) y la condición inicial v implemente el método de colocación ortogonal de Gauss-Radau para el problema de evolución*

$$\begin{cases} u_t = u_{xx} + f(u),\ x \in (0,1),\ t \in (0,T) \\ u_r(0,t) = 0, u(1,t) = b(t) \\ u(x,0) = u_0 \end{cases}$$

Las variables de salida deben ser la discretización temporal t, los puntos de colocación r y los valores de la solución en los puntos del mallado (x,t) dados por la matriz u y la matriz P.

Para resolver el PVI que aparece al implementar el método de colocación,

$$U_t = MU + F(U), \qquad U(0) = (u_0(r_i))$$

utilizar el método de Euler mejorado, que viene dado por la fórmula:

$$U^{j+1} = U^j + \frac{h_t}{2}[G(U^j) + G(U^j + h_t G(t_j, U^j))] \tag{8}$$

donde $G(U) := MU + F(U)$ y h_t es el paso de la discretización temporal.

Práctica 31. *Crear el fichero tipo función* **micolevolnol.m** *que tomando como datos la función f, N (número de puntos de colocación interiores), T, N_t (tamaño de la discretización temporal) y la condición inicial v implemente el método de colocación ortogonal de Gauss-Lobatto para el problema de evolución*

$$\begin{cases} u_t = u_{xx} + f(x,u),\ x \in (-1,1),\ t \in (0,T) \\ u(-1,t) = u(1,t) = 0 \\ u(x,0) = u_0 \end{cases}$$

Las variables de salida deben ser la discretización temporal t, los puntos de colocación r y los valores de la solución en los puntos del mallado (x,t) dados por la matriz u y la matriz P.

Para resolver el PVI que aparece al implementar el método de colocación,

$$U_t = MU + F(U), \qquad U(0) = (u_0(r_i))$$

utilizar el método de Euler mejorado, ver (8)*.*

Práctica 32. *Crea el fichero tipo script* **testmicolevolnol.m** *que ejecute* **micolevolnol.m** *para el problema anterior con las funciones $f(x,u) = \alpha(u - u^3)$ para distintos valores de $\alpha > 0$. Se puede dibujar la solución en 3D y la solución en tiempo final*

```
xx=-1:0.1:1;
U=zeros(size(xx,2),Nt+1);

for j=1:Nt+1
    coef=P*u(:,j);
```

```
    coef=flipud(coef); % cambia un vector [a1;a2;..;aN] en [aN;..;a2;a1]
    U(:,j)=polyval(coef,xx); % evalua un polinomio en un mallado
end

figure(1)
% dibuja en 3D
surf(t,xx,U)
title(['Solucion numerica calculada con ', num2str(size(xx,2)),...
' puntos de mallado.'])
xlabel('Tiempo t')
ylabel('Distancia x')
colorbar

% El perfil de la solucion en tiempo final
figure(2)
plot(xx,U(:,end))
title(['Solucion en t = ',num2str(t(end))])
```

Práctica 33. *Opcionalmente, crea el fichero tipo script* **testmicolevolnol2.m** *que ejecute* **micolevolnol.m** *y dibujer dinámicamente la solución obtenida, realizando la "película" de N_t fotografías de forma que en la fotografía j se pinte la función $u(t_j, x)$. De esta forma se ve dinámicamente la evolución del sistema desde la condición inicial dada por u_0 hasta el estado final dado por $u(T, x)$. Para esto se pueden utilizar los siguientes comandos:*

```
figure(3)

minU=min(U);
minU=min(minU);
maxU=max(U);
maxU=max(maxU);

mov = avifile('moviefuncchainf.avi','fps',20)

for j=1:Nt
    plot(xx,U(:,j))
    axis([-1,1, minU,maxU])
    F(j) = getframe;
    mov = addframe(mov,F(j));
end

title('f=@funcchainf','Fontsize',14)
mov = close(mov);
```

Práctica 34. *Resolver el problema concreto*

$$\begin{cases} c_t - c_{rr} = -\tau^2 R(c,\Theta), & r \in (0,1), t \in [0,T] \\ c_r(0,t) = 0 \\ c(1,t) = 1 \\ c(r,0) = 1{,}01 \end{cases}$$

donde $R(c,\Theta) = (1-c)e^{\gamma-\gamma/\Theta}$ *y toma* $\Theta(r,t) = 1+\alpha(1-c(r,t))$*. Utilizar los valores de* $\gamma = 20$*,* $\tau = 0{,}5$*,* $\alpha = 0{,}6$*,* $T = 0{,}05$*,* $N = 6$*, y* $N_t = 40$*. Crear un fichero tipo función con el nombre* **funcreacarrhenius3.m** *para la función de dicha EDP.*

Práctica 35. *Resolver*

$$\begin{cases} \Theta_t - \Theta_{rr} = \alpha' R(c,\Theta), \quad r \in (0,1), t \in [0,T] \\ \\ c_t - c_{rr} = \alpha R(c,\Theta), \quad r \in (0,1), t \in [0,T] \\ \\ \Theta_r(0,t) = c_r(0,t) = 0 \\ \Theta_r(1,t) + Bi_m\Theta(1,t) = Bi_m\Theta_w(t), \qquad c_r(1,t) = 0 \\ \Theta(r,0) = \Theta_0 \quad c(r,0) = c_0 \end{cases}$$

donde $R(c,\Theta) = (1-c)e^{\gamma-\gamma/\Theta}$*. Cf. el libro de B. Finlayson "Nonlinear Analysis in Chemical Engineering"pp 192 y ss. Utilizad los valores de* $\alpha = 0{,}3$*,* $\alpha' = 0{,}2$*,* $\gamma = 20$*,* $Bi_m = 1$*,* $\Theta_w(t) = 0{,}92$*, y comparad con la solución obtenida en la p. 195. ¿Podrías modificar tu programa para cubrir esas condiciones de contorno?. Utilizad ahora* $Bi_m = 20$*,* $\Theta_w(t) = 1$*, y comparad con la solución obtenida en las p. 201-202.*

Práctica 36. *Resolver el problema concreto*

$$\begin{cases} M_1\Theta_t - \frac{1}{r^d}(r^d\Theta_r)_r = \alpha\tau^2 R(c,\Theta), \quad r \in (0,1), t \in [0,T] \\ \\ M_2 c_t - \frac{1}{r^d}(r^d c_r)_r = -\tau^2 R(c,\Theta), \quad r \in (0,1), t \in [0,T] \\ \\ \Theta_r(0,t) = c_r(0,t) = 0 \\ \Theta(1,t) = c(1,t) = 1 \\ \Theta(r,0) = 1{,}05 \quad c(r,0) = 1{,}0 \end{cases}$$

donde $R(c,\Theta) = ce^{\gamma-\gamma/\Theta}$*. Utilizar los valores de* $d = 0$*,* $M_1 = 176$*,* $M_2 = 199$*,* $\gamma = 20$*,* $\alpha = 0{,}6$*,* $\tau = 0{,}5$*. Comparad la solución obtenida con las del libro de B. Finlayson "Nonlinear Analysis in Chemical Engineering"pp 206 y ss.*

Modifica ahora las condiciones de contorno a

$$\Theta_r(1,t) + Bi_m\Theta(1,t) = Bi_m, \qquad c_r(1,t) + Bi\, c(1,t) = Bi$$

toma $Bi = 27{,}65$*,* $Bi_m = 33{,}25$*,* $\Theta(r,0) = 1{,}1$*. y comparad la solución obtenida con las de la p 208.*

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 5
Resolución Numérica de Ecuaciones Diferenciales Ordinarias

Práctica 37. *El programa Matlab tiene integrados una serie de funciones que permiten resolver* **Ecuaciones Diferenciales Ordinarias (EDO)**. *Los nombres de estos ficheros funcion son:* **ode45.m**, **ode23.m**, **ode113.m**, **ode15s.m** *y* **ode23s.m**. *La sintaxis basicas de estas funciones es:*

```
[t,u]=odeXXX(@func,[tinic  tfin],x0)
```

donde

@func

es el nombre del fichero función de la ODE que se quiere resolver, [tinic tfin] *es el intervalo de tiempo en el que se resuelve la ecuación y* x0 *es el dato inicial de la EDO como vector columna. Por ejemplo (veáse la* **Práctica 39***)*

```
[t,u]=ode15s(@funcreactor,[0  10], [1;1]),
```

resuelve el sistema de un reactor tubular mediante el método **ode15s** *en el intervalo de tiempo* [0,10] *y empezando con el dato inicial* [1;1]. *La expresión de la función*

@funcreactor

está dada en un fichero función (que debe llamarse **funcreactor.m***).*

Es importante tener en cuenta que la salida de las funciones **odeXXX** *consite en* [t,u] *donde t es un vector columna de dimensión N (la discretización del intervalo de tiempos) y u es una matriz de dimensiones $N \times d$ donde d es la dimensión del problema.*

i) Crea un fichero tipo script con nombre **testode.m** *que ejecute algunos de los ficheros* **odeXXX.m** *y pinte*

a) Si la EDO es escalar: la gráfica de la solución aproximada.

b) Si la EDO es en $\mathbb{R}^2$ o en $\mathbb{R}^3$: la gráfica de todas las componentes en la misma ventana y, tras una pausa, pinte en otra ventana la trayectoria de la solución .

Práctica 38. *Crea un fichero tipo función con MATLAB, con nombre* **mirk4.m**, *que implemente el Método de Runge-Kutta de orden 4, evaluando la función de la EDO de un fichero externo. Incluye esta función como uno de los posibles métodos para resolver una EDO en el fichero* **testode.m**. *Compara este método con los ya integrados en Matlab.*

Práctica 39. *Para cada uno de los problemas siguientes explora, usando los métodos descritos arriba, el comportamiento de las soluciones variando la variable independiente en un intervalo apropiado, para varias elecciones de los datos iniciales.*

i) **Reactor Tubular Adiabático:**

$$u'(t) = D_a\, u\, exp\left[\frac{\delta\alpha(1-u}{1+\alpha(1-u)}\right],$$

con condición inicial $u(0) = 1$ eligiendo apropiadamente los coeficientes D_a, δ, α. (ver Davies p. 44)

ii) **Reactor Tubular:**

$$\begin{cases} u'(t) = -D_a\, u\, exp\left[\delta(1-\frac{1}{\Theta})\right] \\ \\ \Theta'(t) = \alpha D_a\, u\, exp\left[\delta(1-\frac{1}{\Theta})\right] - H_w(\Theta - \Theta_w), \end{cases}$$

eligiendo apropiadamente los coeficientes $D_a, \delta, \alpha, H_w, \Theta_w$ *y con datos iniciales* $u(0) = \Theta(0) = 1$.

Notación: *Los ficheros conteniendo las funciones de estas EDOs deben llamarse* **funcreactoradb.m y funcreactor.m** *respectivamente.*

Práctica 40. *Se pueden utilizar para resolver el PVI que aparece al implementar métodos de colocación en problemas de evolución, ver las* **Prácticas 30 y 34**.

```
clear all; close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global  gamma alpha Thiele delta M1 M2 N  b

gamma=20; alpha=.6; Thiele=.5; delta=1; N=10; b=1;

f=@funcolevolarrhenius;

d=0;%d=1;d=2;
r=legendregaussradau(N,d);
[Q,P,A,B]=mismatricesradau(N,r);
M=B+d*diag(1./r)*A;
M1=M(1:N,1:N);
M2=M(1:N,N+1);

u0=1.01+0*r(1:N);

tfin=.2;
intervalt=[0,tfin];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[t,u]=ode15s(f,intervalt,u0);
u=u';
u=[u;b+0*t'];



xx=0:0.01:1;
Nt=size(u,2);
U=zeros(size(xx,2),Nt);
for j=1:Nt
     coef=P*u(:,j);
     coef=flipud(coef); %cambia un vector [a1;a2;..;aN] en [aN;..;a2;a1]
     %tambien existe el comando fliplr(d) que cambia
     %vectores fila [a1,a2,..,aN] en [aN,..,a2,a1]
```

```
    coef = [zeros(1,N+1);coef']; % potencias pares
    coef = coef(:);              % intercala los ceros en el vector columna
    U(:,j)=polyval(coef,xx); %evalua un polinomio en un mallado
end

i=0;
i=i+1; figure(i)
% dibuja en 3D
surf(t,xx,U)
title(['Solucion numerica calculada con N=', num2str(N)])
xlabel('Tiempo t')
ylabel('Distancia x')
colorbar

% El perfil de la solucion en tiempo final
i=i+1; figure(i)
plot(xx,U(:,end))
title(['Solucion en t = ',num2str(t(end))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function G=funcolevolarrhenius(t,u)
global Thiele gamma alpha delta N b M1 M2

T=1+alpha*(1-u);
f=-(Thiele.^2)*(1-u).*exp(gamma-gamma./T);

G=M1*u+f+b*M2;
```

Práctica 41. *Busca alguna ecuación importante e interesante de la Ingeniería Química y resuélvela numéricamente utilizando el fichero* **testode.m***.*

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 6
*** Resolución Numérica de Problemas de contorno**

Práctica 42. *El programa Matlab tiene integrado el programa* **bvp4c.m** *que permite resolver numéricamente* **Problemas de contorno**

$$u''(x) = f(x, u(x), u'(x)), \qquad x \in (a, b)$$
$$bc(u(a), u'(a), u(b), u'(b)) = 0$$

utilizando métodos de colocación. La sintaxis básica de esta función es:

```
sol=bvp4c(@odefun,@bcfun,solinit)
```

donde

```
@odefun
```

es el nombre del fichero función de la EDO que se quiere resolver, como un sistema de dos EDO

```
dv=func(x,v)
```

x *es escalar,* v=$[v_1; v_2] = [u(x); u'(x)]$, dv=$[f_1; f_2] = [u'(x); f(x, u(x), u'(x))]$
donde

```
@bcfun
```

es el nombre del fichero función de la condición de contorno

```
bc= bcfun(ua,ub)
```

debe devolver un vector columna bc=$[bc_1; bc_2]$ *con datos de frontera tipo general, por ejemplo* $ua(1) + \alpha ua(2) - \beta$ *representa una condición de frontera mixta no homogénea*

$$u(a) + \alpha \frac{\partial u}{\partial \nu}(a) = \beta$$

solinit *es el dato inicial de la EDO en un conjunto de nodos ordenados, i.e.* $a = x_1 < x_2 < \cdots < x_N = b$, $u_1, u_2, \cdots u_N = b$ *donde* u_j *es una aproximación inicial de la solución* u *en el nodo* x_j. *Se puede usar la función* bvpinit *que forma una aproximación inicial*

```
solinit= bvpinit(xmesh,uinit)
```

para un mallado inicial xmesh*, o bien usar un fichero de función*

```
solinit= bvpinit(0:0.1:1,@initfun)
```

La solución se evalua en los puntos xint *usando la salida* sol *de* bvp4c *y la función* deval

```
uint= deval(sol,xint)
```

La salida sol *es una estructura con*
sol.x – *mallado seleccionado por* bvp4c
sol.y – *aproximación a* $u(x)$ *en los puntos del mallado* sol.x
sol.yp – *aproximación a* $u'(x)$
sol.solver – *'bvp4c'*

Se pueden considerar problemas singulares del tipo

$$v' = D * v/x + f(x,v), \qquad x \in (0,b)$$
$$bc(v(0), v(b)) = 0$$

con $v = [v_1; v_2]$*. La matriz constante* D *se especifica como el valor del término 'Singular Term' en las opciones de* bvpset *y la función* `@odefun` *evalua sólamente* $f(x,v)$*. Las condiciones de contorno deben ser consistentes con la condición necesaria* $D * v(0) = 0$ *y el dato inicial debe verificar esta condición.*

Práctica 43. *Crea un fichero tipo script con nombre* **testmibvp4c.m** *que*

1.- guarde los datos del problema,

2.- ejecute el fichero **bvp4c.m** *y*

3.- pinte la gráfica de la solución aproximada.

Ejemplo 1 *para la* **Práctica 20** *con* $d = 0$*. Escribir los ficheros tipo función* **odefuncreac.m***,* **bcdirichlet.m** *con la ecuación y las condiciones de contorno de (2).*

Fichero **odefuncreac1.m**

```
function f=odefuncreac1(x,u)
global Thiele k m n
f=zeros(2,1);
f(1)=u(2);
f(2)=(Thiele.^2)*(u(1).^m)./(1+k*u(1).^n);
```

Fichero **bcdirichlet.m**

```
function res=bcdirichlet(ua,ub)
res=zeros(2,1);
res(1)=ua(2);
res(2)=ub(1)-1;  %CC Dirichlet
```

Fichero **testmibvp4c.m**

```
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global Thiele k m n Bim
Thiele=1; k=0; m=2; n=0;

    odefun=@odefuncreac1;
    bcfun=@bcddirichlet;
    solinit=bvpinit([0 .25 .5 .75 1],[1 0]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sol = bvp4c(odefun,bcfun,solinit);
xint = linspace(0,1);
uint = deval(sol,xint);
plot(xint,uint(1,:));
title(['la grafica de la solucion aproximada con f= ', func2str(odefun)])
xlabel('x');
ylabel('solucion u');
```

Ejemplo 2 *para la* **Práctica 21** *de datos iniciales generados por un fichero función.*
Fichero **initfuncreac1.m**

```
function yinit = initfuncreac1(x)
global Thiele
yinit = [  cosh(Thiele*x)/cosh(Thiele)
           Thiele*sinh(Thiele*x)/cosh(Thiele) ]
```

Modificaciones al Fichero **testmibvp4c.m**

```
initfun=@initfuncreac1;
solinit = bvpinit(0:.1:1,initfun);
```

Ejemplo 3 *para la* **Práctica 20** *con* $d = 2$.
Modificaciones al Fichero **testmibvp4c.m**

```
d=2;
options = bvpset('singularterm',[0 0;0 -d]);
sol = bvp4c(odefun,bcfun,solinit,options);
```

Ejemplo 4 *para la* **Práctica 22** *con* $d = 2$.
Modificaciones al Fichero **testmibvp4c.m**

```
d=2;
options = bvpset('singularterm',[0 0 0 0; 0 -d 0 0; 0 0 0 0; 0 0 0 -d]);
sol = bvp4c(odefun,bcfun,solinit,options);
```

DEPARTAMENTO DE MATEMATICA APLICADA
UNIVERSIDAD COMPLUTENSE DE MADRID
Métodos Numéricos en Ingeniería Química
PRACTICAS. Hoja 7
*** Resolución Numérica de Problemas de evolución**

Práctica 44. *El programa Matlab tiene integrado el programa* **pdepe.m** *que permite resolver numéricamente* **Problemas de evolución** *del tipo*

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial t}=x^{-d}\frac{\partial}{\partial x}\left[x^{d}f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right]+s\left(x,t,u,\frac{\partial u}{\partial x}\right),\qquad x\in(x_l,x_r)\qquad t\in(t_0,T)$$

$$u(x,t_0)=u_0(x),\qquad x\in(x_l,x_r)$$

$$p(x,t,u)+q(x,t)f\left(x,t,u,\frac{\partial u}{\partial x}\right)=0,\quad x=x_l,\ x=x_r\quad t\in(t_0,T)$$

También se pueden resolver sistemas de ecuaciones. La sintaxis básica de esta función es:

```
sol=pdepe(d,@pdefun,@icfun,@bcfun,xmesh,tspan)
```

donde

`@pdefun`

es el nombre del fichero función de la EDP que se quiere resolver,

[c,f,s]=pdefun $\left(x,t,u,\frac{\partial u}{\partial x}\right)$

`@icfun`

es el nombre del fichero función dato inicial

u_0=icfun (x)

`@bcfun`

es el nombre del fichero función dato de frontera

$[p_l,q_l,p_r,q_r]$=bcfun (x_l,u_l,x_r,u_r,t)

xmesh *es el vector* $[x_0,x_1,\cdots,x_N]$ *del mallado espacial de puntos* $x_l=x_0<x_1<\cdots<x_N=x_r$
tspan *es el vector* $[t_0,t_1,\cdots,T]$ *especificando el mallado temporal* $t_0<t_1<\cdots<T$

Práctica 45. *Crea un fichero tipo script con nombre* **testmipdepe.m** *que*

1.- guarde los datos del problema,
2.- ejecute el fichero **pdepe.m** *y*
3.- pinte
a) Si la EDP es una ecuación: la gráfica en 3D de la solución aproximada.
b) Si la EDP es un sistema de dos ecuaciones: la gráfica en 3D de cada componente de la solución aproximada.

Ejemplo 1 *para la* **Práctica 35**

Crea los ficheros **pdefuncreac1.m**, **pdefuncreac1ic.m** *y* **pdefuncbcdirichlet.m**
Fichero **testmipdepe.m**

```
close all
clear all
global gamma alpha Thiele
gamma=20; alpha=.6; Thiele=0.5;
    pdefun=@pdefuncreac1;
    icfun=@pdefuncreac1ic;
    bcfun=@pdefuncreac1bc

 d=0;%d=1;d=2;
 xmesh=0:.1:1;
 tspan=0:1:100;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sol = pdepe(d,pdefun,icfun,bcfun,xmesh,tspan);

x=xmesh;
t=tspan;

if size(sol,3)==1
% Extrae la primera componente de la solucion u.
u = sol(:,:,1);

% dibuja en 3D
surf(x,t,u)
title(['Solucion numerica calculada con ', num2str(size(xmesh,2)),...
' puntos de mallado.'])
xlabel('Distancia x')
ylabel('Tiempo t')
colorbar

% El perfil de la solucion en tiempo final
figure
plot(x,u(end,:))
title(['Solucion en t = ',num2str(t(end))])
xlabel('Distancia x')
ylabel(['u(x,',num2str(t(end)),')'])
[UOUT_END,DUOUT_DXEND] = pdeval(d,xmesh,sol(t(end),:,1),xmesh);

%%%%

elseif size(sol,3)==2
u1 = sol(:,:,1);
u2 = sol(:,:,2);

figure
surf(x,t,u1)
title('u_1(x,t)')
```

```
xlabel('Distancia x')
ylabel('Tiempo t')
colorbar

figure
surf(x,t,u2)
title('u_2(x,t)')
xlabel('Distancia x')
ylabel('Tiempo t')
colorbar

end
```